MEMORY ARBITRATION SYSTEM AND METHOD HAVING AN ARBITRATION PACKET PROTOCOL

TECHNICAL FIELD

[001] This present invention is related generally to a memory system for a processor-based computing system, and more particularly, to a hub-based memory system having an arbitration system and method for managing memory responses therein.

BACKGROUND OF THE INVENTION

[002] Computer systems use memory devices, such as dynamic random access memory ("DRAM") devices, to store data that are accessed by a processor. These memory devices are normally used as system memory in a computer system. In a typical computer system, the processor communicates with the system memory through a processor bus and a memory controller. The memory devices of the system memory, typically arranged in memory modules having multiple memory devices, are coupled through a memory bus to the memory controller. The processor issues a memory request, which includes a memory command, such as a read command, and an address designating the location from which data or instructions are to be read. The memory controller uses the command and address to generate appropriate command signals as well as row and column addresses, which are applied to the system memory through the memory bus. In response to the commands and addresses, data are transferred between the system memory and the processor. The memory controller is often part of a system controller, which also includes bus bridge circuitry for coupling the processor bus to an expansion bus, such as a PCI bus.

[003] In memory systems, high data bandwidth is desirable. Generally, bandwidth limitations are not related to the memory controllers since the memory controllers sequence data to and from the system memory as fast as the memory devices allow. One approach that has been taken to increase bandwidth is to increase the speed of the memory data bus coupling the memory controller to the memory devices. Thus, the same amount of information can be moved over the memory data bus in less time. However, despite

increasing memory data bus speeds, a corresponding increase in bandwidth does not result. One reason for the non-linear relationship between data bus speed and bandwidth is the hardware limitations within the memory devices themselves. That is, the memory controller has to schedule all memory commands to the memory devices such that the hardware limitations are honored. Although these hardware limitations can be reduced to some degree through the design of the memory device, a compromise must be made because reducing the hardware limitations typically adds cost, power, and/or size to the memory devices, all of which are undesirable alternatives. Thus, given these constraints, although it is easy for memory devices to move "well-behaved" traffic at ever increasing rates, for example, sequel traffic to the same page of a memory device, it is much more difficult for the memory devices to resolve "badly-behaved traffic," such as bouncing between different pages or banks of the memory device. As a result, the increase in memory data bus bandwidth does not yield a corresponding increase in information bandwidth.

[004] In addition to the limited bandwidth between processors and memory devices, the performance of computer systems is also limited by latency problems that increase the time required to read data from system memory devices. More specifically, when a memory device read command is coupled to a system memory device, such as a synchronous DRAM ("SDRAM") device, the read data are output from the SDRAM device only after a delay of several clock periods. Therefore, although SDRAM devices can synchronously output burst data at a high data rate, the delay in initially providing the data can significantly slow the operating speed of a computer system using such SDRAM devices. Increasing the memory data bus speed can be used to help alleviate the latency issue. However, as with bandwidth, the increase in memory data bus speeds do not yield a linear reduction of latency, for essentially the same reasons previously discussed.

[005] Although increasing memory data bus speed has, to some degree, been successful in increasing bandwidth and reducing latency, other issues are raised by this approach. For example, as the speed of the memory data bus increases, loading on the memory bus needs to be decreased in order to maintain signal integrity since traditionally,

there has only been wire between the memory controller and the memory slots into which the memory modules are plugged. Several approaches have been taken to accommodate the increase in memory data bus speed. For example, reducing the number of memory slots, adding buffer circuits on a memory module in order to provide sufficient fanout of control signals to the memory devices on the memory module, and providing multiple memory device interfaces on the memory module since there are too few memory module connectors on a single memory device interface. The effectiveness of these conventional approaches are, however, limited. A reason why these techniques were used in the past is that it was cost-effective to do so. However, when only one memory module can be plugged in per interface, it becomes too costly to add a separate memory interface for each required memory slot. In other words, it pushes the system controllers package out of the commodity range and into the boutique range, thereby, greatly adding cost.

[006] One recent approach that allows for increased memory data bus speed in a cost effective manner is the use of multiple memory devices coupled to the processor through a memory hub. In a memory hub architecture, or a hub-based memory sub-system, a system controller or memory controller is coupled over a high speed bi-directional or unidirectional memory controller/hub interface to several memory modules. Typically, the memory modules are coupled in a point-to-point or daisy chain architecture such that the memory modules are connected one to another in series. Thus, the memory controller is coupled to a first memory module, with the first memory module connected to a second memory module, and the second memory module coupled to a third memory module, and so on in a daisy chain fashion.

[007] Each memory module includes a memory hub that is coupled to the memory controller/hub interface and a number of memory devices on the module, with the memory hubs efficiently routing memory requests and responses between the controller and the memory devices over the memory controller/hub interface. Computer systems employing this architecture can use a high-speed memory data bus since signal integrity can be maintained on the memory data bus. Moreover, this architecture also provides for easy

expansion of the system memory without concern for degradation in signal quality as more memory modules are added, such as occurs in conventional memory bus architectures.

[008] Although computer systems using memory hubs can provide superior performance, various factors may affect the performance of the memory system. For example, the manner in which the flow of read data upstream (*i.e.*, back to the memory hub controller in the computer system) from one memory hub to another is managed will affect read latency. The management of the flow of read data by a memory hub may be generally referred to as arbitration, with each memory hub arbitrating between local memory read responses and upstream memory read responses. That is, each memory hub determines whether to send local memory read responses first or to forward memory read responses from downstream (*i.e.*, further away from the memory hub controller) memory hubs first. Although the determination of which memory read response has lower priority will only affect the latency of that specific memory read response, the additive effect of the memory read responses having increased latency will affect the overall latency of the memory system. Consequently, the arbitration technique employed by a memory hub directly affects the performance of the overall memory system. Additionally, the implementation of the arbitration scheme will affect the overall read latency as well, since inefficient implementation will negatively impact system memory performance despite utilizing a desirable arbitration scheme. Therefore, there is a need for a system and method for implementing an arbitration scheme for managing memory responses in a system memory having a memory hub architecture.

SUMMARY OF THE INVENTION

[009] A method according to one aspect of the invention includes transmitting a read response on a data path of a memory hub interposed between a transmitting memory hub and a receiving memory hub. The method includes receiving at the memory hub an arbitration packet including data indicative of a data path configuration for an associated read response. The arbitration packet is decoded, and the data path is configured in accordance with the data of the arbitration packet. The associated read response is received

at the memory hub and the associated read response is coupled to the configured data path for transmitting the same to the receiving memory hub.

[010] In another aspect of the invention, a memory hub coupled to at least one memory device is provided. The memory hub includes remote and local input nodes, an output node, and a configurable data path coupled to the remote and local input nodes and further coupled to the output node. The memory hub further includes an arbitration control circuit coupled to the configurable data path, the output node, and the remote input node. The arbitration control circuit generates an arbitration packet for an associated read response coupled through the local input node that includes data indicative of a data path configuration for the associated read response. The arbitration control circuit can further configure the configurable data path in accordance with the data included with an arbitration packet coupled thorough the remote input node in preparation of coupling an associated read response coupled through the remote input node to the output node.

[011] In another aspect of the invention, a memory hub is provided having a bypass data path coupled between an input node and an output node on which read responses are coupled in response to being enabled, and further includes an arbitration control circuit. The arbitration control circuit is coupled to the bypass data path and generates an arbitration packet in response to retrieving read data from a memory device coupled to the memory hub. The arbitration packet has a data path field including activation data to enable a bypass data path of an upstream memory hub. The arbitration control circuit also receives an arbitration packet from a downstream memory hub and enables the bypass data path to couple a read response also received from the downstream memory hub from the input node to the output node.

BRIEF DESCRIPTION OF THE DRAWINGS

[012] Figure 1 is a partial block diagram of a computer system having a memory hub based system memory in which embodiments of the present invention can be implemented.

[013] Figure 2 is a functional block diagram of a arbitration control component according to an embodiment of the present invention that can be utilized in the memory hubs of Figure 1.

[014] Figure 3 is a data structure diagram of a arbitration packet and memory response according to an embodiment of the present invention.

[015] Figure 4 is a flow diagram of the operation of the arbitration control component of Figure 3 according to an embodiment of the present invention

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[016] Figure 4 illustrates a computer system 100 having a memory hub architecture in which embodiments of the present invention can be utilized. The computer system 100 includes a processor 104 for performing various computing functions, such as executing specific software to perform specific calculations or tasks. The processor 104 includes a processor bus 106 that normally includes an address bus, a control bus, and a data bus. The processor bus 106 is typically coupled to cache memory 108, which, is typically static random access memory ("SRAM"). The processor bus 106 is further coupled to a system controller 110, which is also referred to as a bus bridge.

[017] The system controller 110 also serves as a communications path to the processor 104 for a variety of other components. More specifically, the system controller 110 includes a graphics port that is typically coupled to a graphics controller 112, which is, in turn, coupled to a video terminal 114. The system controller 110 is also coupled to one or more input devices 118, such as a keyboard or a mouse, to allow an operator to interface with the computer system 100. Typically, the computer system 100 also includes one or more output devices 120, such as a printer, coupled to the processor 104 through the system controller 110. One or more data storage devices 124 are also typically coupled to the processor 104 through the system controller 110 to allow the processor 104 to store data or retrieve data from internal or external storage media (not shown). Examples of typical storage devices 124 include hard and floppy disks, tape cassettes, and compact disk read-only memories (CD-ROMs).

[018] The system controller 110 contains a memory hub controller 128 coupled to several memory modules 130a-n through a bus system 154, 156. Each of the memory modules 130a-n includes a memory hub 140 coupled to several memory devices 148 through command, address and data buses, collectively shown as bus 150. The memory hub 140 efficiently routes memory requests and responses between the controller 128 and the memory devices 148. Each of the memory hubs 140 includes write buffers and read data buffers. Computer systems employing this architecture allow for the processor 104 to access one memory module 130a-n while another memory module 130a-n is responding to a prior memory request. For example, the processor 104 can output write data to one of the memory modules 130a-n in the system while another memory module 130a-n in the system is preparing to provide read data to the processor 104. Additionally, a memory hub architecture can also provide greatly increased memory capacity in computer systems.

[019] Figure 2 is a functional block diagram illustrating an arbitration control component 200 according to one embodiment of the present invention. The arbitration control component 200 can be included in the memory hubs 140 of Figure 1. As shown in Figure 2, the arbitration control component 200 includes two queues for storing associated memory responses. A local response queue 202 receives and stores local memory responses LMR from the memory devices 148 on the associated memory module 130. A remote response queue 206 receives and stores downstream memory responses which cannot be immediately forwarded upstream through a bypass path 204. An arbitration control circuit 210 is coupled to the queues 202, 206 through a control/status bus 136, which allows the arbitration control circuit 210 to monitor the contents of each of the queues 202, 206, and utilizes this information in controlling a multiplexer 208 to thereby control the overall arbitration process executed by the memory hub 140. The control/status bus 136 also allows "handshaking" signals to be coupled from the queues 202, 206 to the arbitration control circuit 210 to coordinate the transfer of control signals from the arbitration control circuit 210 to the queues 202, 206.

[020] The arbitration control circuit 210 is further coupled to the high-speed link 134 to receive arbitration packets from downstream memory hubs. As will be explained in

more detail below, arbitration packets are provided in advance of an associated memory response, and provide the arbitration control circuit 210 of an upstream memory hub with information to enable the appropriate path through the receiving memory hub in anticipation of receiving the associated memory response. Additionally, the arbitration control circuit 210 generates an arbitration packet to be provided prior to an associated LMR to serve as an early indication of the associated memory response when data is read from the memory devices 148 (Figure 1) in response to a read request. As previously discussed, the arbitration packet will provide upstream memory hubs with appropriate information and give the respective arbitration control circuits 210 time to make decisions regarding enablement of the appropriate data paths before the memory response arrives. The arbitration control circuit 210 prepares the arbitration packet while read data for the memory response is being retrieved from memory devices 148. The arbitration packet is provided through a switch 212 to either the multiplexer 208 or the local response queue 202, depending on whether if the upstream memory hub is idle or busy. The multiplexer 208, under the control of the arbitration control circuit, couples the high-speed link 134 to receive memory responses from the remote response queue 206 or the bypass path 204, arbitration packets from the arbitration control circuit 210, or arbitration packets and memory responses from the local response queue 202. For example, the number and type of data fields of the data structure 300 can be changed or the number of bits for each bit time can be changed and still remain within the scope of the present invention. In an alternative embodiment of the present invention, the arbitration packets are generated in an arbitration packet circuit, rather than in the arbitration control circuit 210, as shown in Figure 2. Additionally, although shown in Figure 2 as providing the arbitration packet to the multiplexer 208 to be injected into the stream of data, the arbitration packet can alternatively be provided to the local response queue 202 and placed before the associated read response packet to be injected into the data stream. It will be appreciated by those ordinarily skilled in the art that modifications to the embodiments of the present invention, such as the location at which the arbitration packet is generated or the manner in which the

arbitration packet is placed into the data stream prior to the associated read packet, can be made without departing from the scope of the present invention.

[021] Figure 3 illustrates a data structure 300 for arbitration packets and memory responses according to an embodiment of the present invention. The data structure 300 is divided into 8-bit bytes of information, with each byte of information corresponding to a sequential bit-time. Each bit-time represents an increment of time in which new data can be provided. A response header field 302 includes two bytes of data that indicate the response is either an arbitration packet or a memory response. An address field 304 includes data that is used to identify the particular hub to which the arbitration packet or memory response is directed. A command code field 306 will have a value to identify the data structure 300 as an arbitration packet, and not as a memory response. Arbitration packets and memory responses are similar, except that the data payload of data fields 308 are "don't cares" for arbitration packets. In the data structure 300, all 16 bits of size fields 310 carry the same value to indicate the size of the data payload carried by the memory response. For example, a "0" indicates that 32 bytes of data are included, and a "1" indicates that 64 bytes of data are included. It will be appreciated by one ordinarily skilled in the art that the embodiment of the data structure 300 shown in Figure 3 has been provided by way of example, and that modifications to the data structure 300 can be made without deviating from the scope of the present invention.

[022] Operation of the arbitration control component 200 (Figure 2) will be described with reference to the flow diagram of Figure 4. Following the receipt of a read data command, at a step 402 the memory hub initiates a read operation to retrieve the requested read data from the memory devices 148 (Figure 1) for the memory response that will be provided to the requesting target. At a step 404, the arbitration control circuit 210 of the memory hub determines whether the local data path is idle by checking the status of the local response queue 202. If the local data path is idle, an arbitration packet is generated by the arbitrations control circuit 210 during the retrieval of the read data from the memory devices 148 at a step 406. When the arbitration packet and the memory response have been prepared, and are ready for transmission, at a step 408 an upstream memory hub is queried

to determine if it is busy. Where the upstream memory hub is idle, the arbitration packet is sent to the upstream memory hub, followed by the memory response at steps 410, 412. However, if the upstream memory hub is busy, the arbitration packet is discarded at a step 414 and the memory response is stored in a local response queue 202 at a step 416. Similarly, in the event that at the step 404 it was determined that the local data path is busy, the memory response is also stored in the local response queue at the step 416. At a step 418 the memory response is stored in the local response queue 202 until it is selected for transmission to the upstream memory hub in accordance with an arbitration scheme implemented by the memory hub. At a step 420, the memory response is transmitted through each upstream memory hub in accordance with the arbitration scheme until the memory response reaches the target destination. Suitable arbitration schemes are well known in the art, and will not be described in detail herein. An example of an arbitration scheme that is also suitable for use is described in more detail in commonly assigned, co-pending U.S. Patent Application No. 10/690,810, entitled ARBITRATION SYSTEM AND METHOD FOR MEMORY RESPONSES IN A HUB-BASED MEMORY SYSTEM to James W. Meyer and Cory Kanski, filed on October 20, 2003, which is incorporated herein by reference.

[023] As described therein, the local and remote response queues 202, 206 and the bypass path 204 are utilized to implement various response arbitration schemes. For example, in one embodiment, the arbitration control circuit executes an arbitration scheme that gives downstream responses, or remote responses, priority over local responses. Alternatively, in another embodiment described, the arbitration control circuit executes an arbitration scheme that gives priority to local responses over downstream responses. In another embodiment, the arbitration control circuit alternates between a predetermined number of responses from local and downstream memory, for example, local and remote responses can be alternately forwarded, or two local responses are forwarded followed by two remote responses, and so on. Another embodiment described therein utilizes an oldest first algorithm in arbitrating between local and downstream memory responses. That is, in operation, the arbitration control circuit 210 monitors response identifier portions of the

memory responses stored in the local response queue and the remote response queue and selects the oldest response contained in either of these queues as the next response to be forwarded upstream. Thus, independent of the response queue in which a memory response is stored, the arbitration control circuit forwards the oldest responses first.

[024] It will be appreciated by those ordinarily skilled in the art that other arbitration methods and schemes can be utilized without departing from the scope of the present invention.

[025] Returning to the steps 410, 412 where the arbitration packet is first transmitted to an upstream memory hub and then followed by the memory response, the arbitration control circuit 210 of the upstream memory hub receives the arbitration packet at a step 422. The arbitration packet is decoded, and the appropriate data path is enabled by the arbitration control circuit 210 based on the information decoded at steps 424, 426. By the time the memory response is received at a step 430, the appropriate data path is enabled by the arbitration control circuit 210. At a step 428, the next upstream memory hub is queried to determine if it is busy. If not, the arbitration packet and then the memory response are transmitted to the next upstream memory hub in a bypass fashion at a step 432. The transmission of the arbitration packet and the memory response in the bypass fashion is facilitated by enabling the appropriate data path through the memory hub based on the decoded information of the arbitration packet that is sent at the step 410 before the associated memory response is sent at the step 412.

[026] Returning to the step 428, if it is determined that the next upstream memory hub is busy, the arbitration packet is discarded at the step 440, and the memory response is stored in the remote response queue 206 until the memory response is selected for transmission to the next upstream memory hub according to the arbitration scheme employed at a step 442. At the step 420, the memory response will make its way upstream through the memory hubs in accordance with the arbitration scheme until reaching its target destination.

[027] From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications

may be made without deviating from the spirit and scope of the invention. For example, embodiments of the present invention have been described herein with respect to a memory hub-based system memory used in a computer system. However, it will be appreciated that embodiments of the present invention can be used in memory systems other than hub-based memory systems, where appropriate. Moreover, embodiments of the present invention can also be used in memory hub-based systems that are utilized in processor based systems, as known in the art, other than computer systems. Accordingly, the invention is not limited except as by the appended claims.